C++ pour la robotique

C++ - Objets suite et fin

Sébastien Rothhut

Message à caractère informatif

Pensez à la lisibilité de votre code :

- Meilleure relecture, par quelqu'un d'autre ou votre vous du futur
- Permet d'éviter les erreurs de syntaxe basiques

Message à caractère informatif

- Adoptez une indentation cohérente

```
int i,j,N;
    cout<<"Tapez la valeur de N : "; cin >> N;

    for(i=0;i<N;i++) {
    for(j=0; j<i; j++) cout<<" ";
    for(; j<N; j++) cout<<"*";
    cout<<endl;
    }
```

Message à caractère informatif

- Adoptez une indentation cohérente

```
int i,j,N;

cout<<"Tapez la valeur de N : ";
cin >> N;

for(i=0;i<N;i++) {
     for(j=0; j<i; j++) cout<<" ";
     for(; j<N; j++) cout<<"*";
     cout<<endl;
}
```

Message à caractère informatif

- N’omettez pas les accolades

```
int i,j,N;

cout<<"Tapez la valeur de N : ";
cin >> N;

for(i=0; i<N; i++) {
      for(j=0; j<i; j++) {
            cout<<" ";
      }
      for(; j<N; j++) {
            cout<<"*";
      }
      cout<<endl;
}
```

Programmation Orientée Objet

Objet : structure de données contenant des attributs et des fonctions

Classe : la définition du type d'un objet

Interface : ensemble des fonctions-membres d'une classe

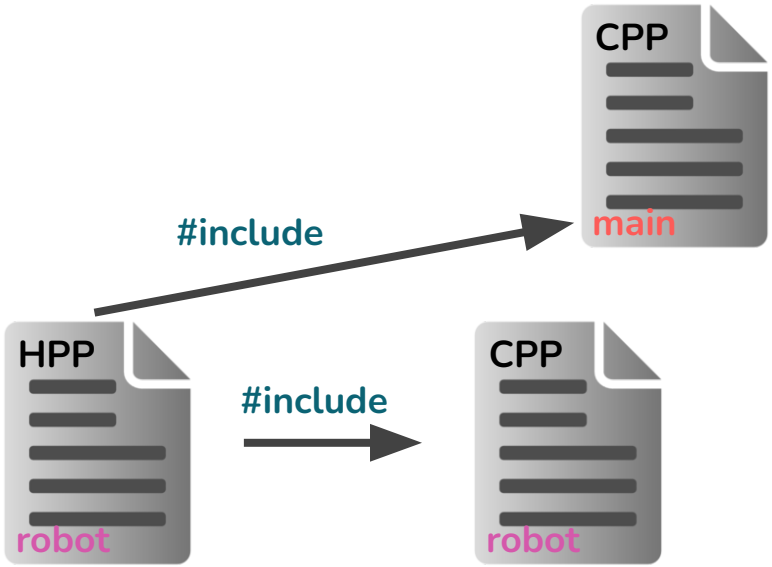
Déclaration d'un enum dans une classe

```
class Cafe {
    public:
        enum Variete {
            ARABICA, ROBUSTA
        };
        Variete variete;
        int quantite;
};
```

Utilisation d'un enum

```
string afficheVarieteCafe(Cafe::Variete variete) {
     switch (variete) {
     case Cafe::ARABICA:
          return "ARABICA";
     case Cafe::ROBUSTA:
          return "ROBUSTA";
     default:
          return "";
     }
}
```

Programmation Orientée Objet



1- Déclaration

2- Définition

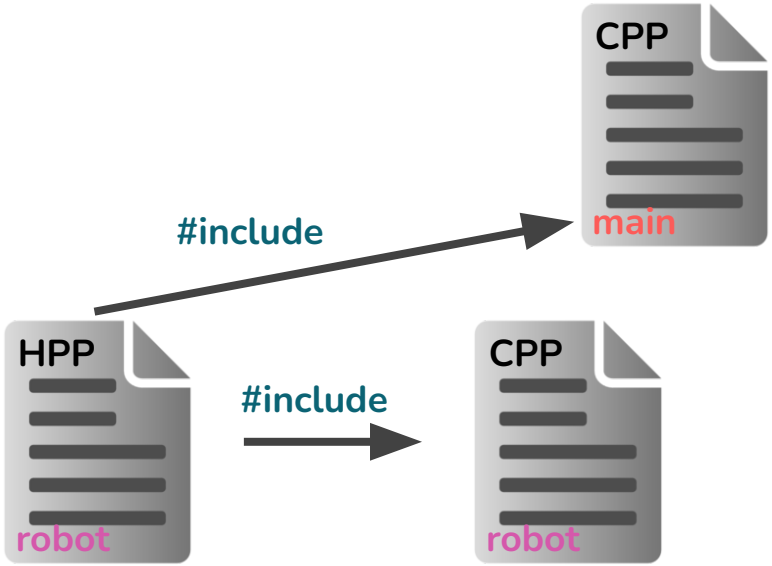
3- Utilisation

Déclaration d'une classe

```
class MachineACafe {
     public:
          MachineACafe();
          MachineACafe(float temp);
          void faireUnCafe(Cafe::Variete variete);
          ~MachineACafe();

     private:
          float niveauEau;
          float temperature;
          const float tempCible;
          bool cafeEnCours = false;
          Cafe reserveCafe;
          void chaufferMachine();
          void faireCoulerCafe(Cafe::Variete variete);
};
```

Définition des fonctions membres



1- Déclaration

2- Définition

3- Utilisation

Définition des fonctions membres

```
MachineACafe::MachineACafe() : tempCible(70.0) {}

MachineACafe::MachineACafe(float temp) : tempCible(temp) {}

MachineACafe::~MachineACafe() {}
```

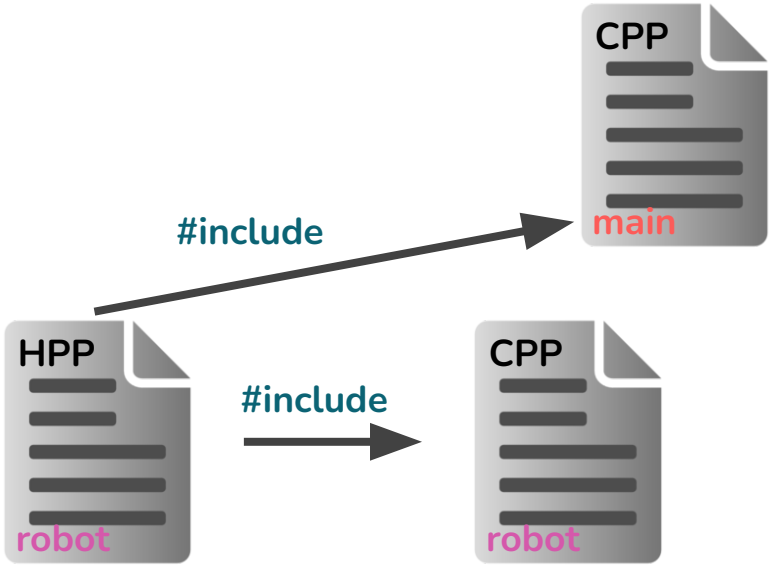
Définition des fonctions membres

```
void MachineACafe::faireUnCafe(Cafe::Variete variete) {
     chaufferMachine();
     faireCoulerCafe(variete);
}

void MachineACafe::chaufferMachine() {
     cout<<"En chauffe..."<<endl;
     cout<<"On atteint "<<tempCible<<" degrés"<<endl;
}

void MachineACafe::faireCoulerCafe(Cafe::Variete variete) {
     cout<<"Je fais couler un "<<afficheVarieteCafe(variete)<<endl;
}
```

Utilisation



1- Déclaration

2- Définition

3- Utilisation

Utilisation

```
int main() {
    MachineACafe maMachine;
    maMachine.faireUnCafe(Cafe::ARABICA);

    return 0;
}
```

Utilisation

```
En chauffe...
On atteint 70 degrés
Je fais couler un ARABICA
```

const correctness

Rappel : le passage d'un paramètre par valeur est un passage par copie de la variable ou de l'objet en paramètre.

```
void afficherVarieteCafe(Cafe cafe) {
     cout<<cafe.variete<<endl;
}
```

Pour passer un objet en paramètre d'une fonction, on préférera faire un passage par référence.

```
void afficherVarieteCafe(Cafe& cafe) {
     cout<<cafe.variete<<endl;
}
```

const correctness

Si un paramètre est passé par référence, il est modifiable par la fonction. Pour empêcher ceci, on ajoute le mot-clef const à l'argument la signature de la fonction.

```
void afficherVarieteCafe(const Cafe& cafe) {
    cout<<cafe.variete<<endl;
}
```

Ceci garantit que le paramètre ne sera pas modifié par la fonction.

const correctness

Pour des variables de types de base, il est inutile de passer par référence, on passera par valeur. La modification est implicitement inopérante.

```
void afficherQuantiteCafe(float quantite) {
    cout<<quantite<<endl;
}
```

Utilisation de const dans les classes

```
class MachineACafe {
    public:
        MachineACafe();
        MachineACafe(float temp);
        void faireUnCafe(Cafe::Variete variete);
        ~MachineACafe();
        const Cafe& getCafe() const;
        void setCafe(const Cafe& cafe);
    private:
        [...]
        Cafe reserveCafe;
        [...]
};
```